FNCE 5676 Lecture 7 – Classification

Two class data

Let's say we can define one class as the "event", like a shot being on goal.

- The **sensitivity** is the *true positive rate* (accuracy on actual events).
- The **specificity** is the *true negative rate* (accuracy on actual non-events, or 1 - *false positive rate*).

Two class data

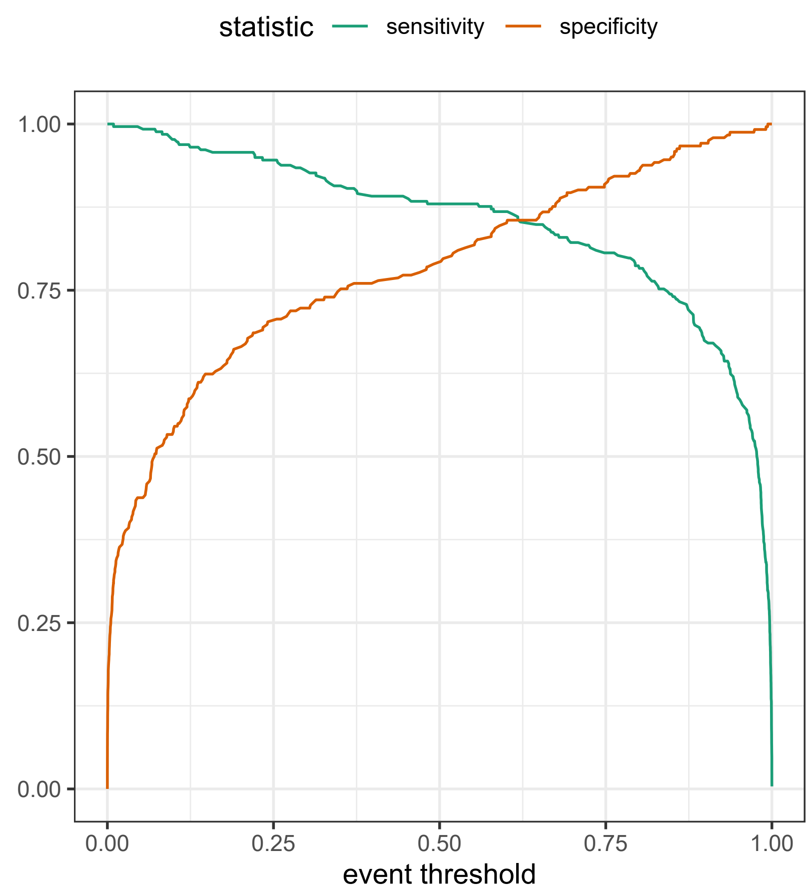
These definitions assume that we know the threshold for converting “soft” probability predictions into “hard” class predictions.
Is a 50% threshold good?
What happens if we say that we need to be 80% sure to declare an event?

- sensitivity ⬇️, specificity ⬆️

What happens for a 20% threshold?

- sensitivity ⬆️, specificity ⬇️

Varying the threshold



ROC curves

To make an ROC (receiver operator characteristic) curve, we:

- calculate the sensitivity and specificity for all possible thresholds
- plot false positive rate (x-axis) versus true positive rate (y-axis)

We can use the area under the ROC curve as a classification metric:

- ROC AUC = 1 💯
- ROC AUC = 1/2 😢

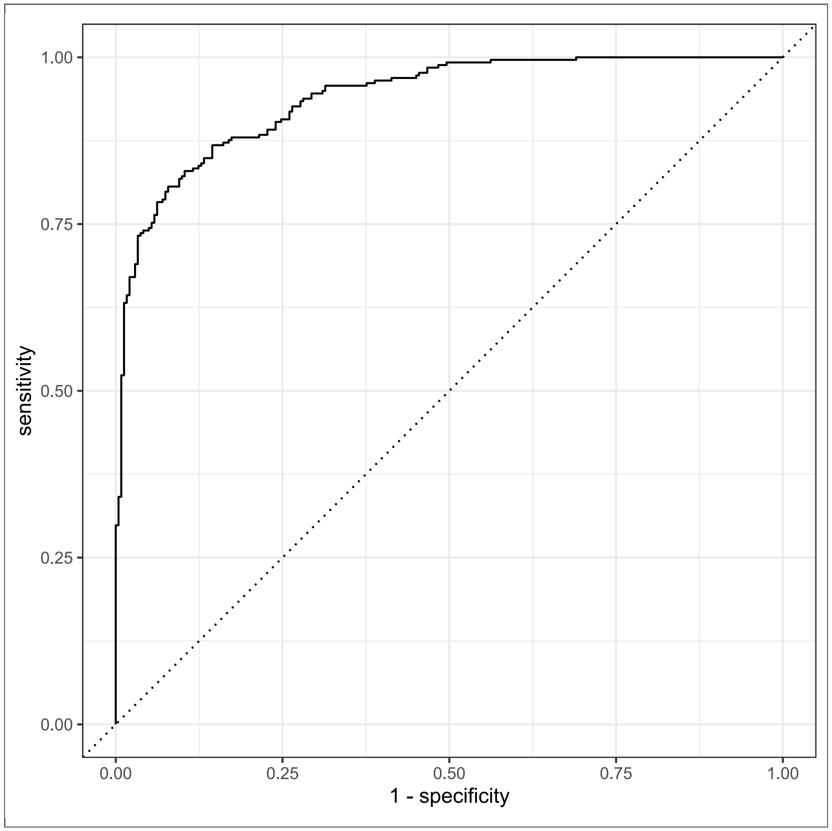
ROC curves

```
# Assumes _first_ factor level is event; there are options to change that
roc_curve_points <- two_class_example %>% roc_curve(truth = truth, estimate = Class1)
roc_curve_points %>% slice(1, 50, 100)
#> # A tibble: 3 × 3
#>   .threshold specificity sensitivity
#>        <dbl>       <dbl>       <dbl>
#> 1 -Inf             0           1
#> 2    0.00236       0.198       1
#> 3    0.0335        0.401       0.996

two_class_example %>% roc_auc(truth = truth, estimate = Class1)
#> # A tibble: 1 × 3
#>   .metric .estimator .estimate
#>   <chr>   <chr>          <dbl>
#> 1 roc_auc binary         0.939
```

ROC curve plot

```
1 autoplot(roc_curve_points)
```



Boosted trees 🌳🌲🌴🌵🌴🌳🌳🌴🌲🌵🌴🌲🌳🌴🌳🌵🌵🌴🌲🌲🌳🌴🌳🌴🌲🌴🌵🌴🌲🌴🌵🌲🌵🌴🌲🌳🌴🌵🌳🌴🌳

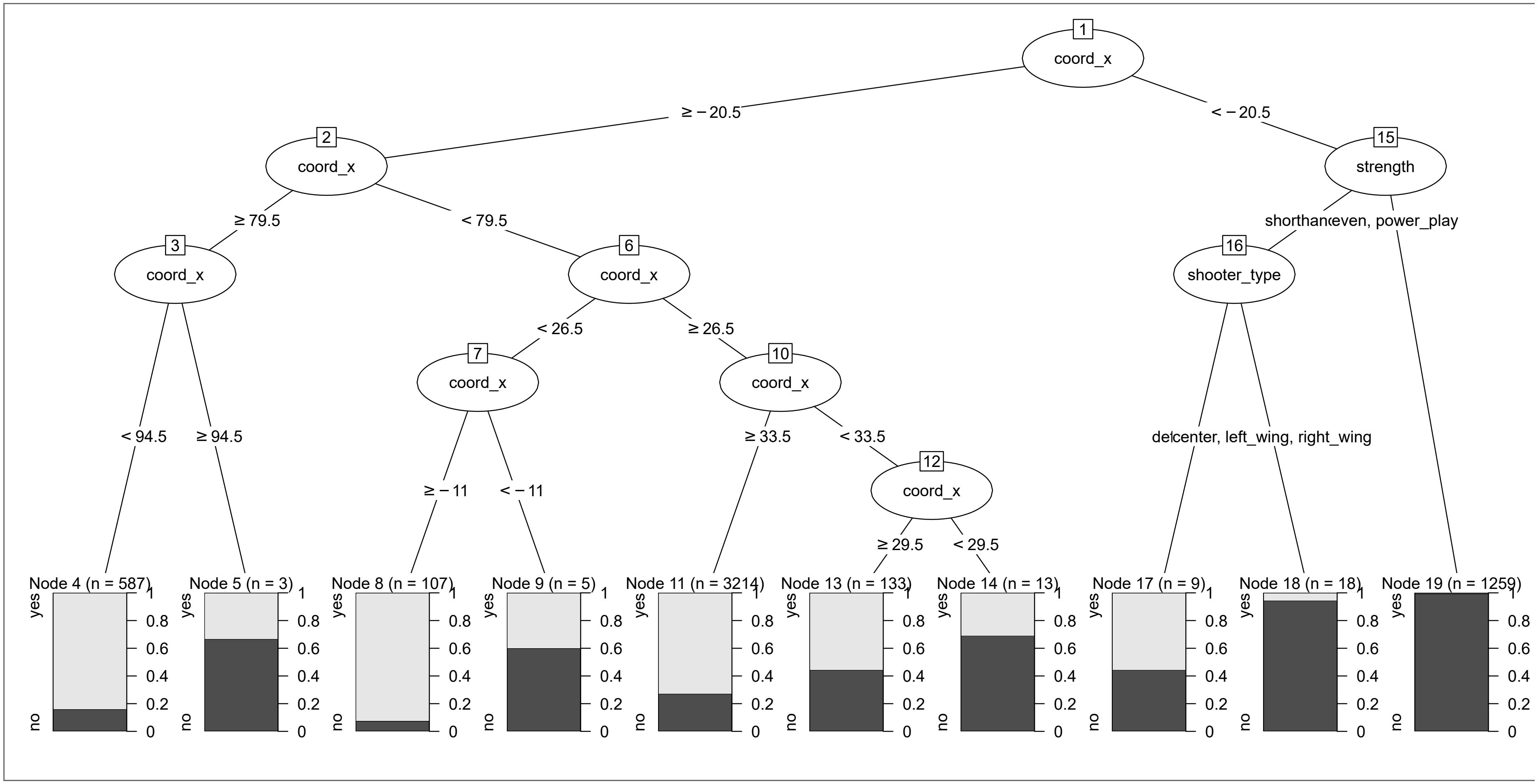
Boosted trees 🌳🌲🌴🌵🌳🌳🌴🌲🌵🌴🌳🌵

- Ensemble many decision tree models

Review how a decision tree model works:

- Series of splits or if/then statements based on predictors
- First the tree *grows* until some condition is met (maximum depth, no more data)
- Then the tree is *pruned* to reduce its complexity

Single decision tree



Boosted trees 🌳🌲🌴🌵🌳🌳🌴🌲🌵🌴🌳🌵

Boosting methods fit a *sequence* of tree-based models.

- Each tree is dependent on the one before and tries to compensate for any poor results in the previous trees.
- This is like gradient-based steepest ascent methods from calculus.

Boosted tree tuning parameters

Most modern boosting methods have *a lot* of tuning parameters!

- For tree growth and pruning (`min_n`, `max_depth`, etc)
- For boosting (`trees`, `stop_iter`, `learn_rate`)

We'll use *early stopping* to stop boosting when a few iterations produce consecutively worse results.

Comparing tree ensembles

Random forest

- Independent trees
- Bootstrapped data
- No pruning
- 1000’s of trees

Boosting

- Dependent trees
- Different case weights
- Tune tree parameters
- Far fewer trees

The general consensus for tree-based models is, in terms of performance: boosting > random forest > bagging > single trees.

Error

×